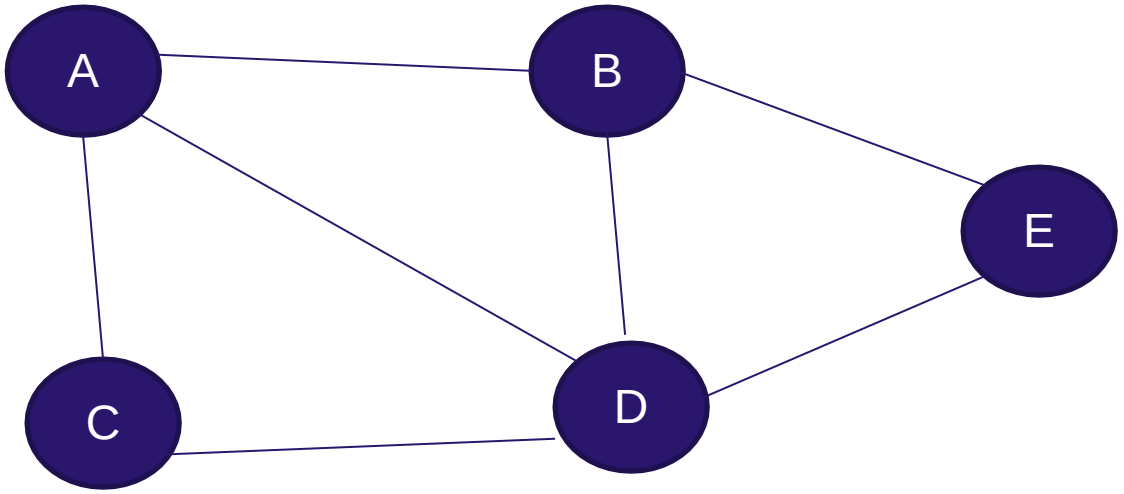
Graph

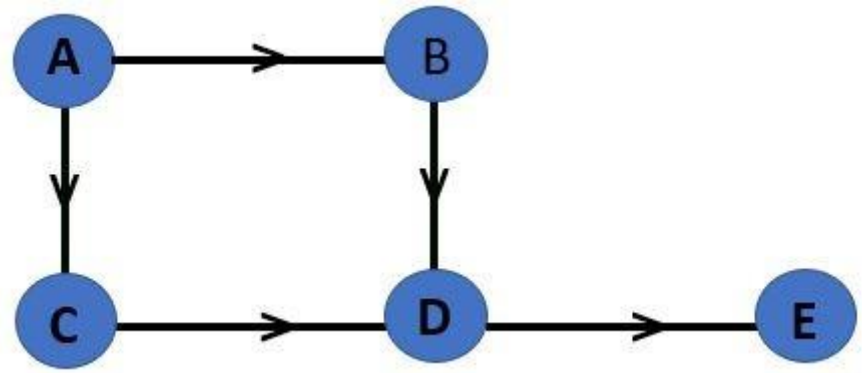
- A graph is a non-linear data structure that consists of two sets i) A set V, called set of vertices (nodes) and ii) A set E, called set of edges (arcs) which connects pair of vertices.

- The following is a graph with 5 vertices and 6 edges. This graph G can be defined as G = (V , E), Where
  - V = {A,B,C,D,E}
  - E = {(A,B),(A,C)(A,D),(B,D),(C,D),(B,E),(E,D)}.



Graph Terminologies

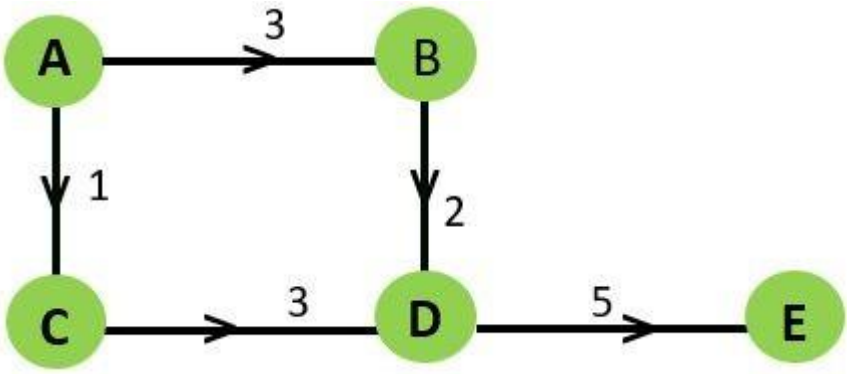
- Digraph : A directed graph is one where the edges can be traversed in a specified direction only.
- In following graph the edges are directional. You can only traverse the edge along its direction.
- In diagraph (Vi, Vj) is different than (Vj, Vi)



Graph Terminologies

- Weighted Graph: A weighted graph is one where the edges are associated with a weight. This is generally the cost to traverse the edge.

- In following graph now the edges have a certain weight associated with them. There are two possible paths between node A to node E.
  Path1 = { AB, BD, DE }, Weight1 = 3+2+5 = 10
  Path2 = { AC, CD, DE }, Weight2 = 1+3+5 = 9
  Clearly, one would prefer Path2 if the goal is to reach node E from node A with minimum cost.

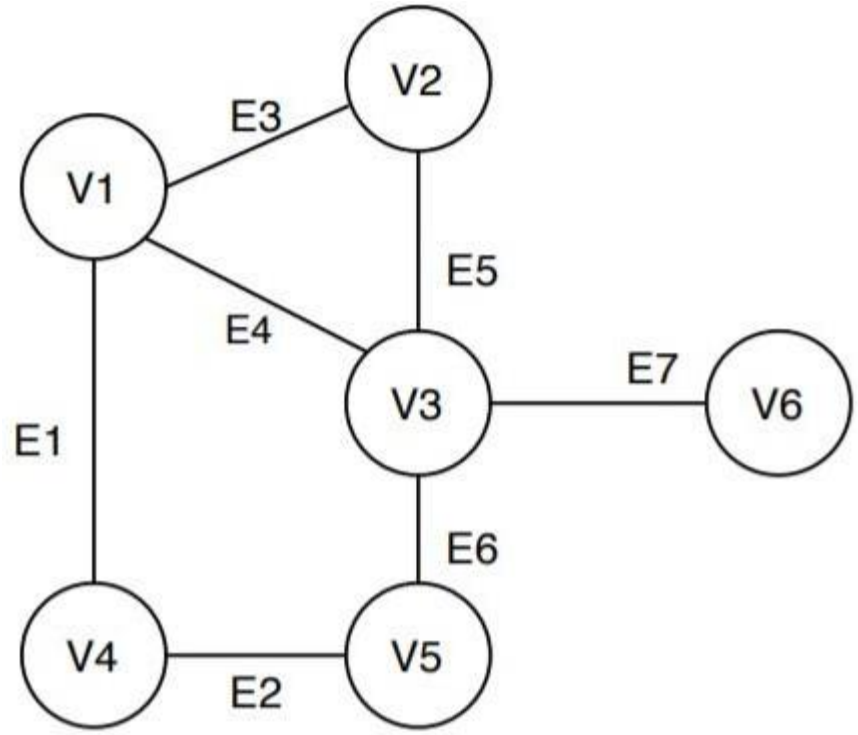


Graph Terminologies

- Adjacent Vertices: A vertex Vi is adjacent to vertex Vj if there is an edge from Vi to Vj.
- Self Loop : If there is an edge whose starting and end vertices are same then it is called a self loop.
- Complete Graph : A graph G is said to be a complete graph if each vertex Vi is adjacent to every other vertex Vj in the graph.
- Acyclic Graph : If a graph does not have any cycle then it is called as acyclic graph. If there is a path containing one or more edges which starts from a vertex Vi, and terminates into the same vertex then path is known as cycle.
- Degree of Vertex : The number of edges connected with vertex Vi is called as the degree of vertex. It is denoted as degree(Vi). Vertex in digraph has indegree and outdegree.

Graph Terminologies

- Connected Graph : In a graph G, two vertices Vi and Vj are said to be connected if there is a path in graph from Vi to Vj. A graph is said to be connected graph if for every pair of distinct vertices Vi and Vj, there is a path.



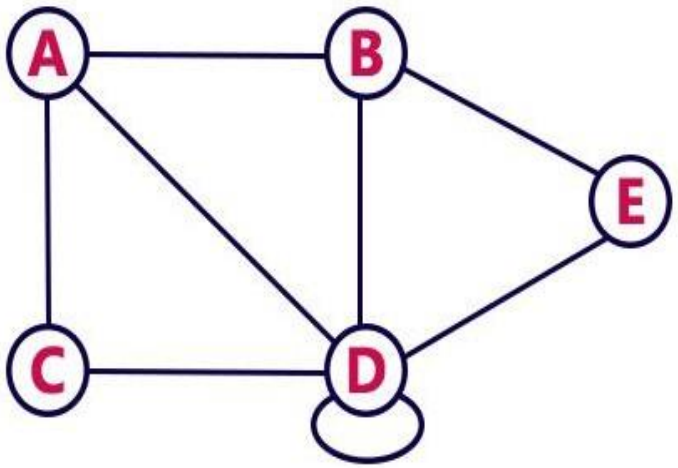
Graph Representation

- Matrix Representation: In this representation, the graph is represented using a matrix of size total number of vertices by a total number of vertices. That means a graph with 4 vertices is represented using a matrix of size 4X4. In this matrix, both rows and columns represent vertices. This matrix is filled with either 1 or 0. Here, 1 represents that there is an edge from row vertex to column vertex and 0 represents that there is no edge from row vertex to column vertex.

- The matrix is known as Adjacency Matrix because an entry stores the information whether two vertices are adjacent or not. It is also called as Bit Matrix or Boolean Matrix or Binary Matrix as the entries are either a 0 or a 1.

Graph Representation

- Matrix Representation:



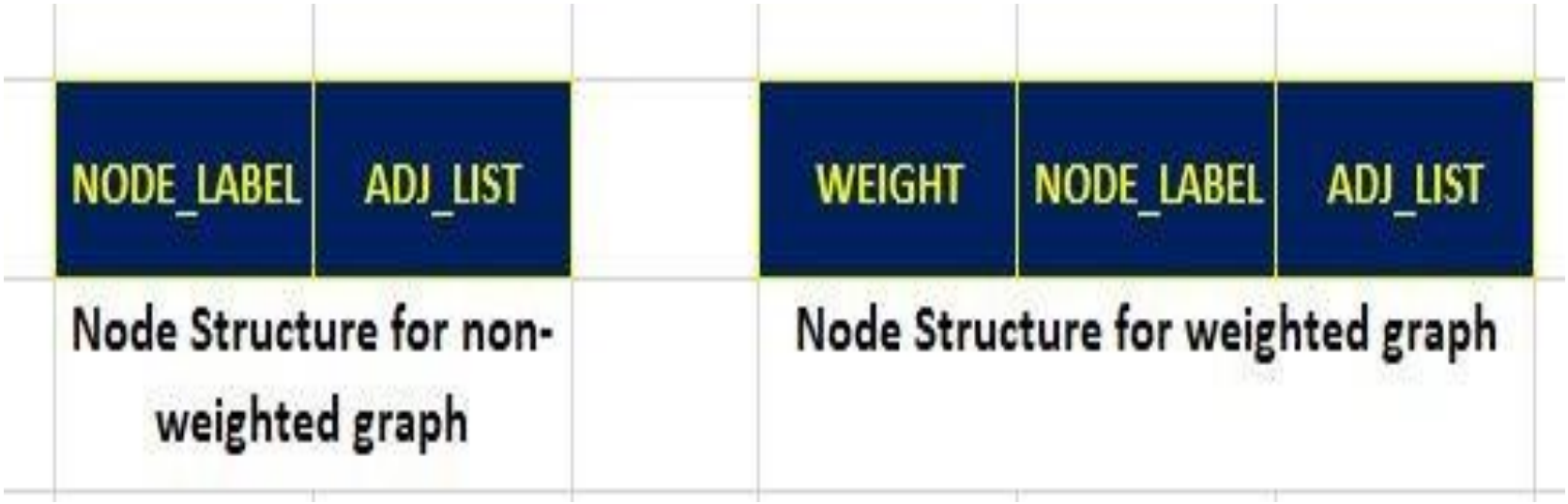
	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

- Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

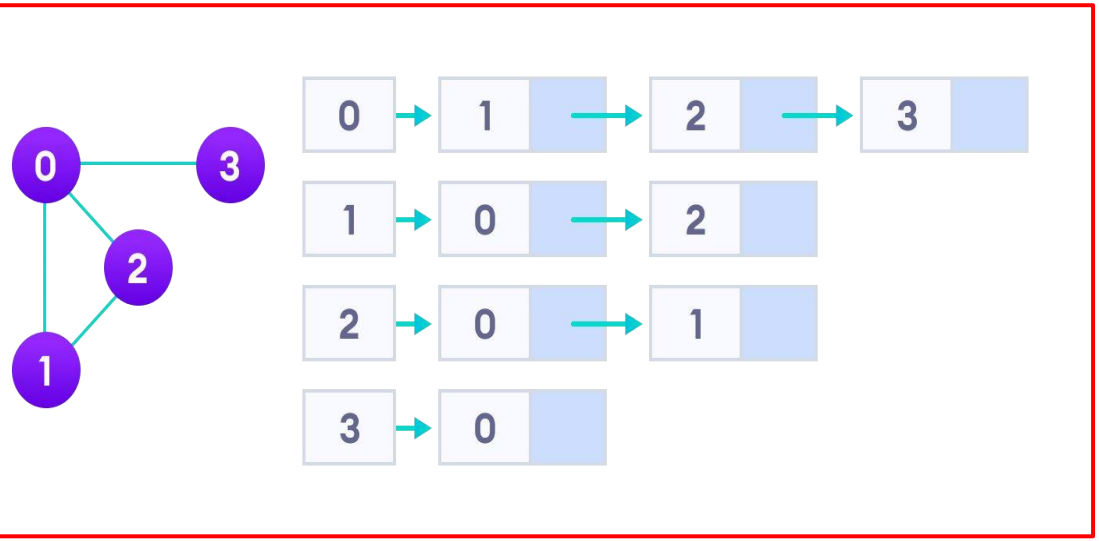
Graph Representation

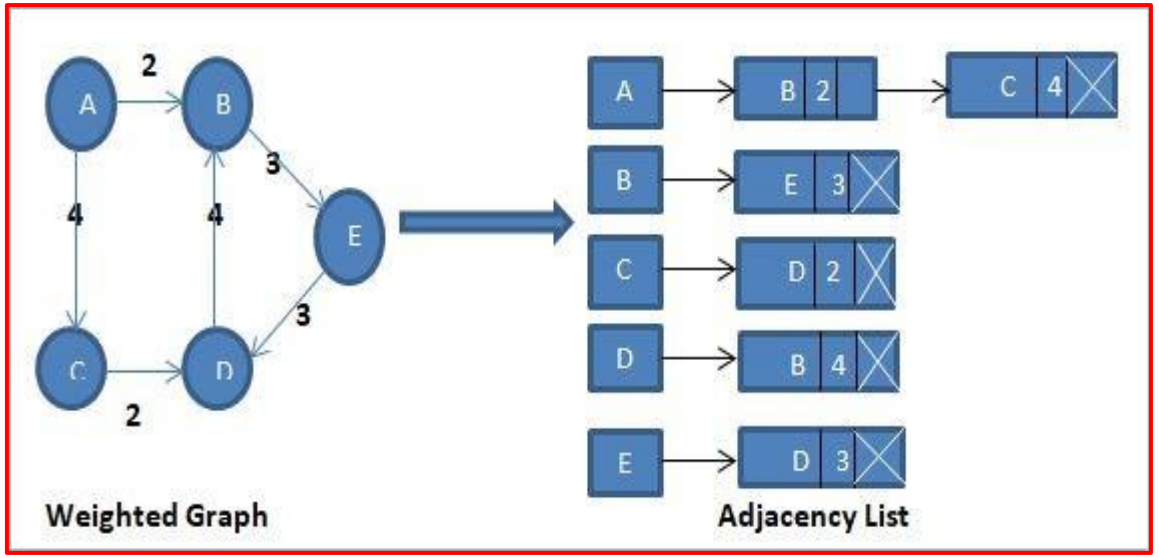
- Linked Representation: In this representation, every vertex of a graph contains list of its adjacent vertices.

- Node Structure



Graph Representation





Graph Traversal

- Graph traversal means visiting every vertex and edge exactly once in a well-defined order. While using certain graph algorithms, you must ensure that each vertex of the graph is visited exactly once. The order in which the vertices are visited are important and may depend upon the algorithm or question that you are solving.

- During a traversal, it is important that you track which vertices have been visited. The most common way of tracking vertices is to mark them.

- Traversal Methods
  - Breadth First Search (BFS)
  - Depth First Search (DFS)

Graph Traversal

- Breadth First Search: The Breadth First Search (BFS) is an algorithm for traversing or searching tree or graph data structures. It explores all the nodes at the present depth before moving on to the nodes at the next depth level.

- The algorithm
  - Pick a node and enqueue all its adjacent nodes into a queue.
  - Dequeue a node from the queue, mark it as visited and enqueue all its adjacent nodes into a queue.
  - Repeat this process until the queue is empty or you meet a goal.

- The program can be stuck in an infinite loop if a node is revisited and was not marked as visited before. Hence, prevent exploring nodes that are visited by marking them as visited.

Graph Traversal

- Breadth First Search Pseudocode :

```
add start vertex to queue q
mark start as visited

while q != empty:
        v = front of queue
        for each adjacent vertex av of v:
                if av is not visited:
                        add av to queue q
                        mark av as visited
```

Graph Traversal

- Depth First Search: Depth First Search is a traversal algorithm is used for traversing a graph. A given path is traversed as long as there is no dead end. Once a dead end is reached, previous vertex is checked for unvisited vertices.

- During the course of the depth first search algorithm, the vertices of the graph will be in one of the two states – visited or initial. Initially, all the vertices are set to initial state. The state of a vertex changes to visited when it is popped from the stack. Initially, the stack is empty.
  - Push or Insert the starting vertex on the stack.
  - Pop or Delete a vertex from the stack.
  - If the deleted vertex is in the initial state, then visit it and change its state to visited. Push all the unvisited vertices adjacent to the popped vertex.
  - Repeat steps 2 and 3 until stack is empty.

Graph Traversal

- Depth First Search:

```
add start vertex to stack st
mark start as visited

while st != empty:
        v = top of stack, pop stack
        for each adjacent vertex av of v:
                if av is not visited:
                        add av to stack st
                        mark av as visited
```